Cheatography

CleanMethods Cheat Sheet

by ChaosJD (ChaosJD) via cheatography.com/99440/cs/21022/

Meaningful Distinction

What's source?

What is destination?

public static void copyChars-(char a1[ ], char a2[ ]) { ... }

Use Pronouncable Names

Write names **out!**

Longer names trumps short Names

Searchable names trumps a constant in Code

Avoid Encodings

Make names pronouncable

Dirty:

class DtaRcrd102 {
private Date genymdhms;
private Date modymdhms;
private final String pszqint = ”102”;}

Hungarian Notation

// Name **not** changed when type changed

PhoneNumber **phone**String;

Method Names

Use	verbs, phrases like PostPayment, deletePage or SavePage
Constr-uctor Overload use	Static factory Methods, with names, that describe the arguments

No Member Prefixes

//Historic => Old Code or old Coder

private String **m**_dsc;

Interface & Implementation

Dirty	Clean
interface IShapeFactory	**class** ShapeFactoryImpl
	interface Shapefactory

Don't Repeat Yourself

Refactor	Clone
	Copy & Paste

Verbs And Keywords

verb/noun Pair:

(to)writeField(name);

assertExpectedEqualsActual(-expected, actual);

Use Descriptive Names

The Sequence should tell a Story:

includeSetup-AndTeardown-Pages();

includeSuiteSetupPage();

includeSetupPages();

Use Problem Domain Names

If is there no Programming technique use the name from the Problem Domain **DomainDrivenDesign** D3 **not** D3.js

Use Solution Domain Names

AccountVisitor	VisitorPattern

Dont' be cute

Dirty	Clean
holyHandGren-ade()	delete-Items()
eatMyShorts()	abort()

One Level of Abstraction per Function

interm ediate Level	String pagePathName = PathParser.render(-pagePath);
low Level	.append(”\n”).

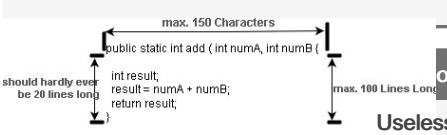
Add Meaningful Context

addressFirstName

addressLastName

addressState

Functions



Sideeffects

If you must have a temporal coupling, you should make it **clear in the name** of the function

checkPasswordAndInitializeS-ession()

Avoid Mental Mapping

Clean:	loop Counter **only:** i, j, k
Dirty:	**k** and **l** because One (1) => too much

cont = continuation

Flag Arguments

Passing a boolean into a funciton is a truly terrible practise

render(boolean isSuite)

Structured Programming

Use	* occasional multiple return,
	* break and
	* continue statement does no harm
Don't USE	**never, ever, any goto statements**
but	only in special languages. Be careful, you can destroy or control flow

on Arguments

Useless:

Niladic(0) **best Solution:**

includeSetupPage()

Polyadic (More then three Argume nts):

ResponseEntity<Map> responseEntity1 = restTemplate.exchange (REST_SERVICE_URI+"/contract/" HttpMethod.PUT, entity, Map.class

Usable

Monadic:

boolean fileExists(“MyFile”)

Dyadic:

assertExpectedEqualsActual(exp-ected, actual);

Triadic:

assertEquals(1.0, amount, .001);

Don't add Gratuitous Context

You need:

* descriptive Skills

* shared cultural backround

* Don't be afraid of Renaming

* Use Refactoring Tools
-> **IntelliJ**

Blocks And Indenting - Arrow Code

The indent level of a function should not be greater then

one or two

TO paragraph

We want every function to be followed by those at the next level of abstraction so that we can read the program, descending one level of abstraction at a time as we read down the list of functions.

To say this differently, we want to be able to read the program as though it were a set of TO paragraphs, each of which is describing the current level of abstraction and referencing subsequent TO paragraphs at the next level down.

MethodHeader:
To include the setups,
Code:
we include the suite setup if this is a suite, then we include the regular setup.

Don't Pun

Use:

insert(), append()

Instead of

add()

Command Query Seperation

Either your function should change the state of an object, or it should return some information about that object:

public boolean set(String attribute, String value);

Clean:

if (attributeExists(”username”)) {setAttribute(”username”, ”unclebob”); … }

Dirty:

if (set(”username”, ”unclebob”))-{...}

Avoid Mental Mapping

loop Counter **only:**	i, j, k
Never	I because One(1)

One Thing

Functions should do **one thing**. They should do it well. They should do it only

* Aids the Reader

* Promote Reuse

*Eases Naming & Testing

*Avoid Sideeffects